Reliability-Driven Energy-Efficient Task Scheduling for Multiprocessor Real-Time Systems

Tongquan Wei, Xiaodao Chen, and Shiyan Hu

***Abstract*—This paper proposes a reliability-driven task scheduling scheme for multiprocessor real-time embedded systems that optimizes system energy consumption under stochastic fault occurrences. The task scheduling problem is formulated as an integer linear program where a novel fault adaptation variable is introduced to model the uncertainties of fault occurrences. The proposed scheme, which considers both the dynamic power and the leakage power, is able to handle the scheduling of independent tasks and tasks with precedence constraints, and is capable of scheduling tasks with varying deadlines. Experimental results have demonstrated that the proposed reliability-driven parallel scheduling scheme achieves energy savings of more than 15% when compared to the approach of designing for the corner case of fault occurrences.**

***Index Terms*—Energy efficient, multiprocessor system, real-time systems, reliability, task scheduling.**

I. Introduction

The number of hardware transient faults has been rising due to the increasing level of integration and reducing size of transistor features in addition to harsh operating environments. The probability of fault occurrences is even higher in a multiprocessor system as a result of large number of components and increased design complexity. Since real-time applications demand both temporal and logical correctness, it is desirable that in the presence of faults real-time tasks finish execution before their respective deadlines. Therefore, real-time embedded systems are typically designed with enough margins to tolerate the worst case expected number of faults by trading off fault coverage and fault detection latency with system performance. Fault tolerance in real-time multiprocessor systems is typically achieved through primary-backup approach where two copies of a task run on different processors [1], [2].

The need for energy-efficient design is increasing for battery-powered real-time systems to reduce power density and enhance the system operational lifetime. Dynamic voltage scaling (DVS) is a popular system level power management technique that exploits technological advances in power supply circuits to reduce processor power consumption by dynamically scaling down the processor speed. Numerous task allocation and scheduling techniques based on DVS have been proposed for energy minimization in multiprocessor systems [3], [4], [5]. However, using DVS technique to reduce power consumption has a negative effect on system reliability. It has been shown that scaling down the processor speed increases the transient faults rates, especially for those induced by cosmic ray radiations, and thus degrades system reliability [6]. Therefore, fault-tolerance and energy are two design constraints that interplay and need to be jointly optimized.

The joint optimization of energy and fault-tolerance as two important design constraints for safety-critical real-time systems has been extensively investigated in the recent past [7], [8], [9]. Both energy savings and fault-tolerance are achieved by utilizing the slack time in a task schedule. However, all these researches focus on the joint optimization of the two design constraints for uni-processor systems. Wei *et al.* [10] proposed an energy-efficient task allocation and scheduling schemes with deterministic fault-tolerance capabilities for symmetric multiprocessor systems executing tasks with hard real-time constraints. However, the presented scheme cannot handle the scheduling of tasks with precedence constraints. In [11], the authors described a flexible multiprocessor platform using modest hardware support to enable an energy-efficient fault-tolerance mechanism. Timeliness of the system is not considered as a design constraint. In [12], the authors addressed the scheduling and voltage scaling for hard real-time applications that have been statically mapped on heterogeneous distributed embedded systems. Tasks in a given task set are assumed to share a common deadline and the effect of voltage scaling on system reliability is taken into account. Similar to [12], Zhu *et al.* [13] also investigated the reliability aware power management for real-time tasks sharing a common deadline, but they assume the investigated multiprocessor system is homogenous and tasks to be scheduled are independent.

This paper proposes a reliability-driven energy-efficient task scheduling scheme for real-time homogeneous or heterogeneous multiprocessor embedded systems that optimizes system energy consumption under stochastic fault occurrences. The proposed multiprocessor task scheduling scheme is featured by a novel fault adaptation variable β that models the uncertainties in fault occurrences. The proposed scheme considers both the dynamic power and the leakage power, is able to handle the scheduling of both independent tasks and tasks with precedence constraints, and is capable of scheduling tasks with varying deadlines.

The rest of this paper is organized as follows. Section II describes the system architecture and models. Section III formulates the energy optimization problem as an integer linear program (ILP). Section IV describes the proposed reliability-driven scheduling scheme that generates energy optimum task schedules. Section V presents the numerical results and Section VI concludes this paper.

II. System Architecture and Models

The focus of this paper is on multiprocessor systems where processing units are tightly coupled, inter-unit communication is achieved via common shared memory, and communication cost is assumed to be negligible. It is assumed that the target multiprocessor system consists of N DVS-equipped processing units and each processing element

Manuscript received April 20, 2011; revised May 13, 2011; accepted May 29, 2011. Date of current version September 21, 2011. This work was supported in part by the Fundamental Research Funds for the Central Universities of China, under the Grant 78220021. This paper was recommended by Associate Editor Y. Xie.

T. Wei is with the Department of Computer Science, East China Normal University, Shanghai 200062, China (e-mail: tqwei@cs.ecnu.edu.cn).

X. Chen and S. Hu are with the Department of Electrical and Computer Engineering, Michigan Technological University, Houghton, MI 49931 USA (e-mail: cxiaodao@mtu.edu; shiyan@mtu.edu).

Digital Object Identifier 10.1109/TCAD.2011.2160178

supports L discrete frequency levels. As part of the future work, this system model can be improved by incorporating communication cost into task execution time. The problem formulation will be slightly modified and the proposed framework could be extended to handle that.

A. Application and Energy Model

Consider a real-time task set Γ consisting of M periodic tasks with precedence constraints: $\{\Gamma|\tau_1, \tau_2, \dots, \tau_M\}$. A task is ready for execution only if the execution of its predecessor is completed. Similarly, the successor of a task is ready for execution only if the execution of the task is completed. The timing characteristics of the task τ_m are defined to be a tuple $\tau_m = \{T_m, D_m, C_m\}$, where T_m is the period, D_m is the deadline, and C_m is the task execution cycles. It is assumed that the period of a task equals its deadline, i.e., $T_m = D_m$.

The power consumption of a CMOS device can be modeled as the sum of dynamic power consumption and static power consumption. The average dynamic power consumption p_d can be estimated by a strictly increasing and convex function, i.e., $p_d \propto f^3$ [14]. Let p_s denote the static power consumption of a device, I_{subn} denote the sub-threshold leakage current, and I_j denote the reverse bias junction current, then the static power consumption of the device is given by $p_s = V_{dd}I_{subn} + |V_{bs}|I_j$, where V_{bs} is the body bias voltage and V_{dd} is the supply voltage [15].

The total energy (E_{tot}) consumed by a multiprocessor system during the execution of real-time tasks in a given task set is hence estimated by

$$E_{tot} = \sum_{m=1}^{M} \frac{C_m}{f_m}(p_d + p_s) \tag{1}$$

where $\frac{C_m}{f_m}$ denotes the execution time of task τ_m at the frequency f_m.

B. Recovery Model

Checkpointing technique is used in this paper to provide fault-tolerance. It is assumed that checkpointing intervals for a given task are equal. Let k_m be the worst case number of fault occurrences during the execution of task τ_m at the frequency level l, f_m be the operating frequency of task τ_m at the frequency level l, and O_m be the optimal number of checkpoints for task τ_m that minimizes the task response time at the frequency level l, then O_m is given by

$$O_m = \left\lceil \sqrt{\frac{k_m C_m}{c_s f_m}} - 1 \right\rceil \quad \text{or} \quad \left\lfloor \sqrt{\frac{k_m C_m}{c_s f_m}} - 1 \right\rfloor$$

where c_s is the checkpointing overhead [8]. Assuming the best case where no faults occur during the execution of task τ_m, then CB_m, which is defined to be the execution cycles of task τ_m with only checkpointing overhead, is expressed as $CB_m = C_m + O_m \times c_s \times f_m$. Assuming the worst case where faults occur at the end of checkpointing saving, then CW_m, which is defined to be the execution cycles of task τ_m including checkpointing overhead and the worst case error recovery overhead, is given by

$$CW_m = CB_m + \frac{k_m C_m}{(O_m + 1)} + 2k_m \times c_s \times f_m.$$

In this equation, $\frac{C_m}{(O_m+1)}$ denotes the checkpoint interval and $\frac{k_m C_m}{(O_m+1)}$ indicates the overhead to recover from k_m fault occurrences. The term $2k_m \times c_s \times f_m$ gives the overheads of k_m checkpoint savings and k_m system state retrievals [8].

CB_m and CW_m of task τ_m are constant for a fixed operating frequency f_m or frequency level l of task τ_m. However, the current execution time C_m of task task τ_m is a random variable due to the stochastic property of fault occurrences. Since it is difficult to solve a mathematical program with uncertainty, the energy optimization problem is transformed into a deterministic optimization problem without random variables. In this paper, a variable β, referred to as fault adaptation variable, is introduced to model the uncertainty in task execution time due to fault occurrences. The current execution time of task τ_m including fault recovery overhead can be expressed as a function of CB_m and CW_m, that is

$$C_m = (1 - \beta) \times CB_m + \beta \times CW_m \tag{2}$$

where $0 \le \beta \le 1$. The execution time of task τ_m is CW_m if $\beta = 1$ and is CB_m if $\beta = 0$.

Let λ_l denote the average fault arrival rate at the frequency level l for $1 \le l \le L$, where L is the number of processor supported frequency levels. The λ_l at frequency f_m can be derived using the equation $\lambda_l = \gamma \times e^{-\alpha f_m}$, where γ and α are constant parameters [6]. Since transient faults are typically modeled using the Poisson distribution, the probability of k_m fault occurrences during the execution of task τ_m at frequency f_m is hence given by

$$\frac{e^{-\lambda_l \frac{C_m}{f_m}} \times (\lambda_l \frac{C_m}{f_m})^{k_m}}{k_m!}.$$

The reliability of a task is defined to be the probability of completing the task successfully subject to faults [6]. As a result, the reliability of task τ_m is the probability of completing the task successfully subject up to k_m faults. The task level reliability is maintained if all tasks in the task set finish the execution successfully under their respective given reliability target. Let RG_m denote the reliability goal of task τ_m and R_m denote the reliability of the task, the reliability of task τ_m is maintained if the inequality

$$RG_m \le R_m = \sum_{k_m} \frac{e^{-\lambda_l \frac{C_m}{f_m}} \times (\lambda_l \frac{C_m}{f_m})^{k_m}}{k_m!} \tag{3}$$

holds for $\beta = 1$. Since λ_l, C_m, f_m, and k_m all are functions of the frequency level l of task τ_m, R_m is also a function of the frequency level l of task τ_m. For a given frequency level l $(1 \le l \le L)$ and $\beta = 1$, λ_l, C_m, and f_m are all known; thus, the worst case number of fault occurrences k_m subject to target reliability RG_m can be iteratively derived using (3).

III. ILP Formulation

The reliability-driven energy-efficient task-to-processor assignment and scheduling is formulated as an integer linear programming problem. The object function is the multiprocessor energy consumption that considers both the dynamic power and leakage power, as is given in (1). Energy optimization is performed under constraints of the task execution time as a function of the fault adaptation variable β for both independent and dependent tasks with varying deadlines.

The variable $A_{m,l,n}$ is introduced to denote the scheduling of task m at the frequency level l on processor n. $A_{m,l,n}$ is equal to 1 if task τ_m is scheduled at the lth frequency level on processor n, and is equal to 0 if τ_m is scheduled at any other frequency levels or on any other processors. For the sake of easy presentation, an ILP definition for single processor task scheduling is given below to demonstrate the formulation. Then the variable $A_{m,l,n}$ becomes $A_{m,l,1}$. Let the variable S_i and S_j denote the start time of task τ_i and task τ_j, respectively, and the variable $Smin_{(i,j)}$ denote the minimum of the S_i and S_j. Thus, the equation $Smin_{(i,j)} = min(S_i, S_j)$ holds for any two tasks in a given task set. The $b_{i,j}$ is a auxiliary binary decision variable indicating the relationship of $Smin_{(i,j)}$, S_i, and S_j. If $S_i < S_j$ holds, i.e., $Smin_{(i,j)} = S_i$, then $b_{i,j} = 1$, else $b_{i,j} = 0$. $\mathcal{H}$ is a large constant number and is set to 10 000 in the experimental section. Similar to $Smin_{(i,j)}$, the variable $Smax_{(i,j)}$ is introduced to indicate the maximum of the S_i and S_j. That is, $Smax_{(i,j)} = max(S_i, S_j)$ holds for any two tasks in a given task set. Two auxiliary variables $h_{i,j}$ and $g_{i,j}$ are also introduced as pseudo-linear constraints to facilitate the formulation.

Given independent or dependent tasks τ_m with respective deadlines of D_m for $m = 1, 2, \ldots, M$, the number of processor supported frequency levels L, and a value of β for $0 \le \beta \le 1$, the goal is to find $A_{m,l,1}$ and S_m for $m = 1, 2, \ldots, M$ and $l = 1, 2, \ldots, L$ that minimize the system energy consumption under the given constraints. The formulation of the ILP problem is thus given as follows:

$$\text{minimize: } E_{tot}(A_{m,l,1}; S_m)$$

$$\text{subject to: } A_{m,l} = 0, 1 \tag{4}$$

$$\sum_{l=1}^{L}\sum_{n=1}^{N} A_{m,l,1} = 1 \tag{5}$$

$$Smin_{(i,j)} \le S_i \tag{6}$$

$$Smin_{(i,j)} \le S_j \tag{7}$$

$$Smin_{(i,j)} \ge S_i - \mathcal{H} \times (1 - b_{i,j}) \tag{8}$$

$$Smin_{(i,j)} \ge S_j - \mathcal{H} \times b_{i,j} \tag{9}$$

$$b_{i,j} = 1, 0 \tag{10}$$

$$Smax_{(i,j)} = S_i + S_j - Smin_{(i,j)} \tag{11}$$

$$h_{i,j} = \mathcal{H} \times (S_i - Smax_{(i,j)}) + S_j \tag{12}$$

$$S_i - h_{i,j} \ge \sum_{l=1}^{L} A_{j,l,1}\left(\frac{C_j}{f_j}\right) \tag{13}$$

$$g_{i,j} = \mathcal{H} \times (S_j - Smax_{(i,j)}) + S_i \tag{14}$$

$$S_j - g_{i,j} \ge \sum_{l=1}^{L} A_{i,l,1}\left(\frac{C_i}{f_i}\right) \tag{15}$$

$$S_i + \sum_{l=1}^{L} A_{i,l,1}\left(\frac{C_i}{f_i}\right) \le D_i \tag{16}$$

$$S_{m_1} < S_{m_2} \tag{17}$$

$$(\tau_{m_1}, \tau_{m_2}) \in \Gamma; 1 \le l \le L.$$

Equation (4) restates the definition of the variable $A_{m,l,1}$, i.e., $A_{m,l,1}$ is either 0 or 1. Equation (5) indicates that each task runs at one and only one processor frequency level. Inequalities (6)–(10) ensure that $Smin_{(i,j)} = min(S_i, S_j)$ holds. Equation (11) identifies the maximum of S_i and S_j. Equations (12)–(15) ensure that the executions of task τ_i and τ_j on the processor have no overlapping. Equation (16) indicates that each task has to finish the execution before its deadline. Equation (17) enforces the precedence constraints assuming task τ_{m_2} cannot start until the execution of task τ_{m_1} finishes. This formulation can be easily extended to handle the scheduling of tasks on multiple processors. Due to space limitation, it is not presented in this paper.

IV. RDPS for Multiprocessor Systems

The reliability-driven energy-efficient task scheduling aims to generate an energy optimum schedule and meet the target task level reliability goal under the assumption of the Poisson probability distribution of fault occurrences. A systematic reliability-driven parallel task scheduling algorithm, motivated from [18] and referred to as RDPS, is proposed in this section to derive the desired fault adaptation variable β and generate the energy optimum task schedule for a given task set with fault-tolerance requirements. As described in Fig. 1, for a given task set Γ and the target reliability RG_m, a value of the fault adaptation variable β is randomly picked, and the associated fault recovery overhead is incorporated in task execution time based on the selected β (step A). The schedule of the task set is then generated by solving the ILP definition (step B). Finally, the reliability R_m of each task is derived using Monte Carlo simulation (step C). If the reliability of a task does not satisfies the stop-condition of the RDPS algorithm, the fault adaptation variable β is adjusted and the above process is repeated. If the reliability of all tasks satisfies the stop-condition of the RDPS algorithm, the output task schedule is optimal in energy consumption and its reliability meets the system reliability requirement. In other words, the output task schedule is the desired task schedule if $(R_m - RG_m) \ge \epsilon > 0$ holds, where ϵ is an arbitrarily small positive number. The following sections describe each step of the RDPS algorithm in details.

A. β-Enabled Parallel Multiprocessor Task Scheduling

One of the key contributions of this paper is to introduce a fault adaptation variable β that adapts task execution time including fault recovery overhead to the Poisson probability distribution of fault occurrences. Unlike the traditional approach of designing for corner cases, this novel technique enables the designing of reliability-driven energy-efficient real-time embedded systems based on status quo of fault occurrences. Since $0 \le \beta \le 1$, the execution time of task τ_m ranges from CB_m to CW_m, as is shown in (2).

```
Require: task set $\Gamma$ and the reliability goal $RG_m$
Ensure: the desired $\beta$ and energy optimum task schedule
1. randomly pick a value of $\beta$ in $0 \le \beta \le 1$
2. repeat
3.     $A$: update $C_m$ of $\tau_m$ based on $\beta$, then update $\beta$
4.     $B$: generate a task schedule by solving the ILP
5.     $C$: derive $R_m$ of task $\tau_m$ using Monte Carlo simulation
6. until $(R_m - RG_m) \ge \epsilon > 0$
```

Fig. 1. RDPS algorithm to iteratively derive the fault adaptation variable β and generate the energy optimum task schedule.

Due to the statistical property of transient fault occurrences, there exists no deterministic relationship between the fault adaptation variable β and the reliability of a task. As a result, the current reliability R_m of task τ_m, if does not meet the stop-condition of the RDPS algorithm, cannot be used to direct the selection of the value of β to be used in the next iteration of the RDPS algorithm. A simple yet efficient binary search-based approach is hence utilized to obtain the next values of β. More specifically, given the initial range of $\beta \in [0, 1]$, and the initial value of β denoted by β_0, the next values of β could be in the range of both $[0, \beta_0]$ and $[\beta_0, 1]$. If the current reliability R_m of task τ_m does not reach the specified reliability goal of RG_m, the next candidates of β that will be used to generate the task schedule are $\beta_1 = \beta_0/2$ and $\beta_2 = (\beta_0+1)/2$. Repeat this process until the R_m is greater than yet close enough to the RG_m.

B. Generate Task Schedules Using a LP Solver and Sequential Rounding Technique

Due to the difficulties to solve ILP program efficiently, linear program (LP) with sequential rounding is utilized to compute task schedules by relaxing the integer constraints of the ILP. LP with sequential rounding is a well-known technique and has been extensively investigated in the literature [16].

The integer constraint of the ILP definition that $A_{m,l,n}$ is either 0 or 1 is relaxed and it becomes a real number such that $0 \le A_{m,l,n} \le 1$. A commercial or open-source LP solver is used to solve the energy optimization problem defined in (1) under the relaxed constraint. The output of the LP solver, $A_{m,l,n} \in \mathbb{R}$ for $m = 1, 2, \ldots, M$, $l = 1, 2, \ldots, L$, and $n = 1, 2, \ldots, N$, indicates that each task can be assigned to more than one processors, and the task assigned to a specific processor can run at one or more frequency levels. This will incur extra overhead owing to task migration among processors and task frequency switching on the same processor. Sequential rounding technique is therefore utilized to assign a task to one and only one processor and to set the operating frequency of the task at one and only one frequency level. More specifically, the variable $A_{m,l,n}$ is rounded to either 0 or 1 by comparing the $A_{m,l,n}$ with a predefined threshold value, which can be adjusted based on time requirements to generate the desired task schedule.

C. Derive Task Reliability Using the Monte Carlo Simulation and Latin-Hypercube Sampling

The reliability of a task in a given task set is evaluated under fault occurrences of Poisson probability distribution using Monte Carlo simulation. Oftentimes, the reliability of a task schedule is obtained in three major steps. In step 1, the reliability evaluation of the current task schedule is started by generating the average fault arrival rate at various frequency levels at which tasks in the given task set are scheduled. In step 2, the number of fault occurrences during the execution of each task is then generated based on the probability distribution of the task, and the execution time of the task is updated to include fault recovery overhead. In step 3, the feasibility of the generated task schedule is verified. Steps 2 and 3 constitute one sample of the Monte Carlo simulation. Repeat steps 2 and 3 to take more than 10 000 Monte Carlo samples, and the reliability of the current task schedule is derived as the ratio of the number of samples where the schedule is feasible to the total number of Monte Carlo samples. If the current reliability is greater than and yet close enough to the target reliability, the resultant task schedule is the desired schedule and the execution of the RDPS algorithm exits. Otherwise, the RDPS algorithm jumps from step C to step A and continues its execution, as is described in Fig. 1.

Although the Monte Carlo simulation described above exhibits relative generality and insensitivity to the number of stochastic fault occurrences, it is expensive for accurate reliability estimation of a task schedule. Therefore, the Latin-Hypercube sampling method is adopted in this paper to improve the efficiency of reliability estimation for a task schedule by sampling fault occurrences more systematically. Two hundred samples of fault occurrences are taken in the proposed scheme for reliability evaluation.

Of the three steps of the RDPS algorithm, steps A and C take constant time. Hence, the computation overhead of the proposed RDPS algorithm mainly depends on the overhead to derive the task schedule using ILP solver in step B. In step B, an interior point technique-based ILP solver is adopted. The complexity of the interior point ILP solver is $O(z^3)$, where z is the total number of variables in the ILP formulation [17] and is given by $m(5m + nl + 1)$. The overall computation overhead of the RDPS algorithm is $O(4mnlz^3)$ for 16 samples of the β. This computation overhead is acceptable for RDPS because it is an offline algorithm.

V. Numerical Results

Extensive experiments were carried out over a simulated multiprocessor system to validate the proposed schemes for energy efficiency and running time. It is assumed that the multiprocessor supports four discrete voltage levels, which are 0.5 V, 0.65 V, 0.8 V, and 1.0 V. The values of the dynamic power and the leakage power of the processor, scaled to 70 nm technology based on the technology scaling trend, is adopted from [15]. The proposed reliability-driven parallel task scheduling algorithm RDPS was implemented in C++, and the simulation was performed on a machine with Intel Core 2 Quad 2.4 GHz processor and 8 GB memory. Task execution times are in the range of 10–40 ms. Transient fault occurrences are assumed to follow the Poisson probability distribution, and the average fault arrival rate at the lowest processor speed is assumed to be 4.

TABLE I

AVERAGE ENERGY CONSUMPTION (IN mJ) OF TASK SETS WITH VARYING SIZES AND THE AVERAGE CPU TIME (IN SECONDS) OF THE PROPOSED RDPS ALGORITHM ($0 < \beta < 1$) FOR TARGET RELIABILITY OF 0.99

Task Set	1-Core					2-Cores					4-Cores				
	$\beta = 0$	$0 < \beta < 1$ (RDPS)			$\beta = 1$	$\beta = 0$	$0 < \beta < 1$ (RDPS)			$\beta = 1$	$\beta = 0$	$0 < \beta < 1$ (RDPS)			$\beta = 1$
Size	E_B	E_R	CPU_4	E_{WR}	E_W	E_B	E_R	CPU_4	E_{WR}	E_W	E_B	E_R	CPU_4	E_{WR}	E_W
10–40	158	187	29.1	19.1%	231	152	185	32.2	17.1%	223	165	192	87.1	21.6%	245
41–70	273	298	36.2	35.4%	461	268	292	49.3	37.9%	470	271	316	102.7	31.7%	463
71–100	521	590	45.7	20.1%	738	513	577	71.2	24.0%	759	542	595	172.3	20.9%	752

Three designing approaches, i.e., the best case, the worst case, and the proposed stochastic RDPS approach, are compared in energy savings and computational complexity under a given task level target reliability. Let E_B, E_W, and E_R denote the energy consumption of a task set under the best case fault occurrences ($\beta = 0$), the worst case fault occurrences ($\beta = 1$), and the stochastic fault occurrences ($0 < \beta < 1$), respectively. E_R in fact indicates energy consumptions of the proposed RDPS algorithm. Then let $E_{WR} = \frac{(E_W - E_R)}{E_W} \times 100\%$ denote energy savings of the proposed RDPS scheme ($0 < \beta < 1$) when compared to the approach of designing a system under the worst case of fault occurrences ($\beta = 1$).

Table I shows the average energy consumption of task sets with varying sizes for a common target reliability of 0.99. Tasks in a task set are assigned to a 1-core, 2-core, and 4-core system, respectively, for both $\beta = 0$ and $\beta = 1$. For the proposed RDPS algorithm ($0 < \beta < 1$), tasks in a task set are always assigned to all four cores of the Core 2 Quad processor due to the scheduling parallelism property of the scheme. The proposed parallel RDPS algorithm ($0 < \beta < 1$) achieves energy savings of up to 37% when compared to the approach of designing for the worst case faults ($\beta = 1$). For instance, when tasks of a given task set the size of which is 41–70 are assigned to a 2-core system, the RDPS scheme ($0 < \beta < 1$) consumes 37.9% less energy when compared to the designing approach for the worst case faults ($\beta = 1$).

The proposed RDPS algorithm can efficiently compute the desired task schedule in parallel. The notation CPU_4 is introduced to denote the CPU time the RDPS algorithm takes to compute the desired task schedule using all 4-cores of the Core 2 Quad processor. Table I also gives the average CPU time of the proposed RDPS algorithm ($0 < \beta < 1$) for a common target reliability of 0.99. The CPU time of the RDPS algorithm for task sets with 10–100 tasks is in the order of seconds. For example, the average CPU time for the task set with 41–70 tasks assigned to a 2-core system is 49.3 s.

VI. CONCLUSION

This paper proposed a novel RDPS scheme that is featured by a fault adaptation variable β. The RDPS algorithm optimizes system energy consumption under stochastic fault occurrences and accelerates the computing of the desired task schedule by utilizing techniques such as the β-enabled scheduling parallelism, sequential rounding, and Latin-Hypercube sampling-based Monte Carlo simulation. Experimental results have shown that the proposed parallel RDPS scheme achieved energy savings of more than 15% when compared to the approach of designing for the corner case of fault occurrences.

REFERENCES

[1] D. Mosse, R. Melhem, and S. Ghosh, "Analysis of a fault-tolerant multiprocessor scheduling algorithm," in *Proc. 24th Int. Symp. Fault Tolerant Comput.*, Jun. 1994, pp. 16–25.

[2] S. Ghosh, R. Melhem, and D. Mosse, "Fault-tolerance through scheduling of aperiodic tasks in hard real-time multiprocessor systems," *IEEE Trans. Parallel Distribut. Syst.*, vol. 8, no. 3, pp. 272–284, Mar. 1997.

[3] K. Gururaj and J. Cong, "Energy efficient multiprocessor task scheduling under input-dependent variation," in *Proc. DATE*, Apr. 2009, pp. 411–416.

[4] C. Xian, Y. Lu, and Z. Li, "Energy-aware scheduling for real-time multiprocessor systems with uncertain task execution time," in *Proc. DAC*, Jun. 2007, pp. 664–669.

[5] G. Zeng, T. Yokoyama, H. Tomiyama, and H. Takada, "Practical energy-aware scheduling for real-time multiprocessor systems," in *Proc. 15th IEEE Int. Conf. Embedded Real-Time Comput. Syst. Applicat.*, Aug. 2009, pp. 383–392.

[6] D. Zhu, R. Melhem, and D. Mosse, "The effects of energy management on reliability in real-time embedded systems," in *Proc. Int. Conf. Comput.-Aided Des.*, 2004, pp. 35–40.

[7] R. Melhem, D. Mosse, and E. Elnozahy, "The interplay of power management and fault recovery in real-time systems," *IEEE Trans. Comput.*, vol. 53, no. 2, pp. 217–231, Feb. 2004.

[8] Y. Zhang and K. Chakrabarty, "A unified approach for fault tolerance and dynamic power management in fixed-priority real-time embedded systems," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 25, no. 1, pp. 111–125, Jan. 2006.

[9] D. Zhu, H. Aydin, and J. Chen, "Optimistic reliability aware energy management for real-time tasks with probabilistic execution times," in *Proc. Real-Time Syst. Symp.*, Nov.–Dec. 2008, pp. 313–322.

[10] T. Wei, P. Mishra, K. Wu, and H. Liang, "Fixed-priority allocation and scheduling for energy-efficient fault-tolerance in hard real-time multiprocessor systems," *IEEE Trans. Parallel Distribut. Syst.*, vol. 19, no. 11, pp. 1511–1526, Nov. 2008.

[11] M. Rashid, E. Tan, M. Huang, and D. Albonesi, "Power-efficient error tolerance in chip multiprocessor systems," *IEEE Micro*, vol. 25, no. 6, pp. 60–70, Nov.–Dec. 2005.

[12] P. Pop, K. Poulsen, V. Izosimov, and P. Eles, "Scheduling and voltage scaling for energy/reliability trade-offs in fault-tolerant time-triggered embedded systems," in *Proc. 5th IEEE/ACM Int. Conf. Hardware/Software Codes. Syst. Syn.*, Sep. 2007, pp. 233–238.

[13] X. Qi, D. Zhu, and H. Aydin, "Global reliability-aware power management for multiprocessor real-time systems," in *Proc. IEEE Int. Conf. Embedded Real-Time Comput. Syst. Applicat.*, Aug. 2010, pp. 183–192.

[14] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design: A System Perspective*. Reading, MA: Addison-Wesley, 1992.

[15] R. Jejurikar, C. Pereira, and R. Gupta, "Leakage aware dynamic voltage scaling for real-time embedded systems," in *Proc. IEEE Des. Automat. Conf.*, Jul. 2004, pp. 275–280.

[16] S. Shah, A. Srivastava, D. Sharma, D. Sylvester, D. Blaauw, and V. Zolotov, "Discrete Vt assignment and gate sizing using a selfsnapping continuous formulation," in *Proc. ACM/IEEE Int. Conf. Comput.-Aided Des.*, Nov. 2005, pp. 705–712.

[17] L. Behjat and A. Chiang, "Fast integer linear programming based models for VLSI global routing," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2005, pp. 6238–6243.

[18] C. Liao and S. Hu, "Multi-scale variation-aware techniques for high performance digital microfluidic lab-on-a-chip component placement," *IEEE Trans. Nanobiosci.*, vol. 10, no. 1, pp. 51–58, Mar. 2011.